Presenter –
MUSKAN RATHORE
(DATA AND APPLIED SCIENTIST, MICROSOFT)

NLP – WORD EMBEDDINGS

WORD MEANING

Word is a symbol that signifies something

Symbol ←→ Signifies

Symbol –"Chair" ←→ Signifies 

Definition of "meaning" of a word:

- the idea that is represented by a word, phrase, etc.
- the idea that a person wants to express by using words, signs, etc.
- the idea that is expressed in a work of writing, art, etc.

WORD NET

e.g., synonym sets containing "good":

```
from nltk.corpus import wordnet as wn
poses = { 'n':'noun', 'v':'verb', 's':'adj (s)', 'a':'adj', 'r':'adv'}
for synset in wn.synsets("good"):
    print("{}: {}".format(poses[synset.pos()],
            ", ".join([l.name() for l in synset.lemmas()])))
```

```
noun: good
noun: good, goodness
noun: good, goodness
noun: commodity, trade_good, good
adj: good
adj (sat): full, good
adj: good
adj (sat): estimable, good, honorable, respectable
adj (sat): beneficial, good
adj (sat): good
adj (sat): good, just, upright
…
adverb: well, good
adverb: thoroughly, soundly, good
```

e.g., hypernyms of "panda":

```
from nltk.corpus import wordnet as wn
panda = wn.synset("panda.n.01")
hyper = lambda s: s.hypernyms()
list(panda.closure(hyper))
```

```
[Synset('procyonid.n.01'),
Synset('carnivore.n.01'),
Synset('placental.n.01'),
Synset('mammal.n.01'),
Synset('vertebrate.n.01'),
Synset('chordate.n.01'),
Synset('animal.n.01'),
Synset('organism.n.01'),
Synset('living_thing.n.01'),
Synset('whole.n.02'),
Synset('object.n.01'),
Synset('physical_entity.n.01'),
Synset('entity.n.01')]
```

DRAWBACKS OF THESAURUS

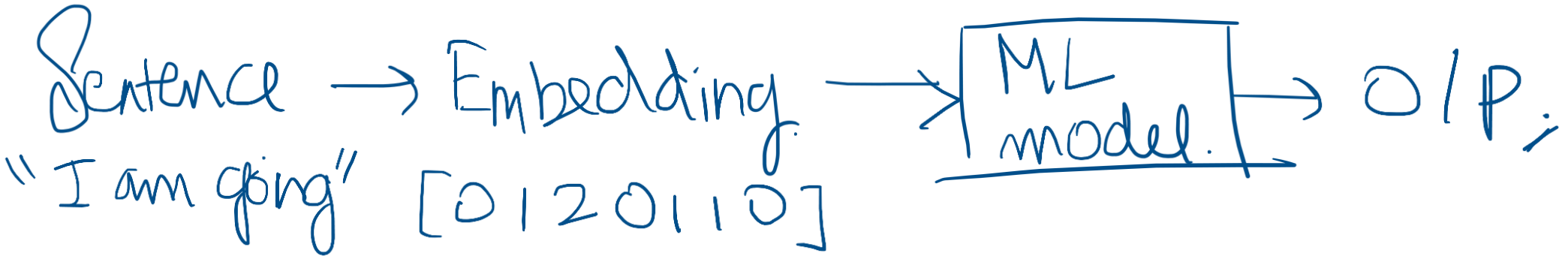
- A useful resource but missing nuance:

• e.g., "proficient" is listed as a synonym for "good" - This is only correct in some contexts

- Missing new meanings of words: • e.g., wicked, badass, nifty, wizard, genius, ninja, bombest
- Impossible to keep up-to-date!
- Subjective
- Requires human labor to create and adapt
- Can't be used to accurately compute word similarity

SENTENCE EMBEDDING

- For classification and regression ML problems, based on sentence inputs such as Sentiment Classification, Email Spam classification, Predicting review rating etc
- Convert sentence to its numeric representation and use this sentence vector as an input to any ML model normally like you would for any dataset.



1. **Remove Punctuation and Special Characters**: Eliminate non-alphanumeric characters.
2. **Tokenization**: Split the text into individual words or tokens.
3. **Lowercasing**: Convert all characters to lowercase.
4. **Stopword Removal**: Filter out common words (e.g., "the", "is", "and").
5. **Lemmatization or Stemming**: Reduce words to their base or root form.
6. **Handle Numerical and Date Values**: Standardize numerical values and dates.
7. **Handle HTML Tags and URLs**: Remove or replace HTML tags and URLs.
8. **Customized Cleaning Steps (if required)**: Address specific cleaning needs.

DATA PREPROCESSING

BAG OF WORDS (BOW)

- **Process**: It involves creating a vocabulary of unique words from the text corpus and counting the occurrence of each word in each document.
- **Representation**: Each document is represented as a sparse vector where each dimension corresponds to a word from the vocabulary, and the value at each dimension represents the count of that word in the document.
- Implement using scikit-learn's **CountVectorizer** implementation

John likes to watch movies
Mary likes movies too
John also likes football

	John	likes	to	watch	movies	Mary	too	also	football
Sentence 1	1	1	1	1	1	0	0	0	0
Sentence 2	0	1	0	0	1	1	1	0	0
Sentence 3	1	1	0	0	0	0	0	1	1

TF-IDF (TERM FREQUENCY – INVERSE DOCUMENT FREQUENCY)

- TF-IDF is a statistical measure used to evaluate the importance of a word in a document relative to a collection of document

  - **Term Frequency (TF):**
    $$\mathrm{TF}(t,d) = \frac{\text{count of term } t \text{ in document } d}{\text{total terms in document } d}$$

  - **Inverse Document Frequency (IDF):**
    $$\mathrm{IDF}(t,D) = \log\left(\frac{\text{total documents in corpus } |D|}{\text{number of documents containing term } t}\right)$$

  - **TF-IDF Score:**
    $$\text{TF-IDF}(t,d,D) = \mathrm{TF}(t,d) \times \mathrm{IDF}(t,D)$$

ONE HOT REPRESENTATION OF WORDS

- Consider words as unique symbols.
- Localist representation using one hot vectors.

Ex: Cat -> [0 0 0 0 0 0 0 0 1 0 0], Dog -> [0 0 1 0 0 0 0 0 0 0 0], Giraffe -> [0 0 0 0 0 0 0 0 0 0 1]

Do you see an issue with this representation??

ONE HOT REPRESENTATION OF WORDS

- Consider words as unique symbols.
- Localist representation using one hot vectors.

Ex: Cat -> [0 0 0 0 0 0 0 0 1 0 0], Dog -> [0 0 1 0 0 0 0 0 0 0 0], Giraffe -> [0 0 0 0 0 0 0 0 0 0 1]

- Issues:
  - Infinite number of words in English
  - Each word can create many other words by adding "ing", "ally", "est" etc.
  - Size of one hot vector to represent words = number of total words ; impossible vector – infinite size

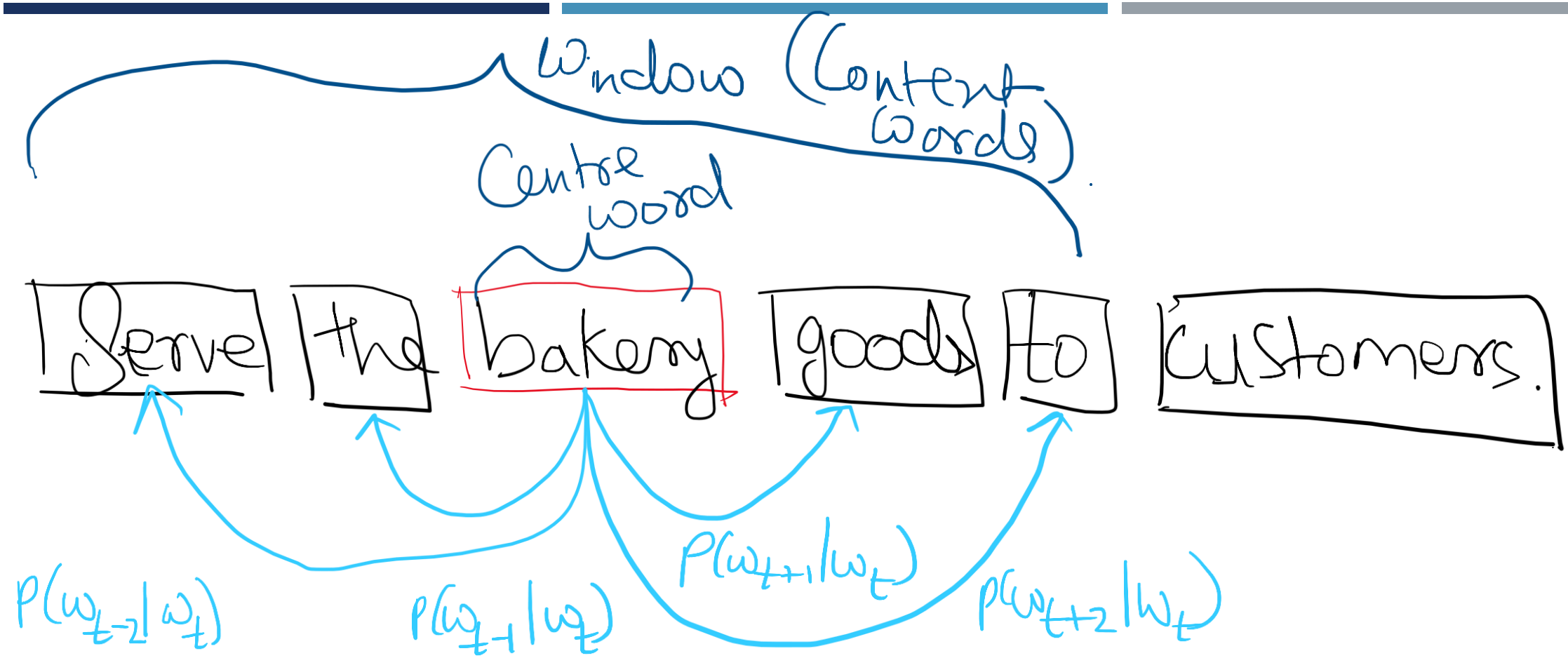
- A meaning of word can be given by using the words it appears with.
- "You shall know a word by the company it keeps" – J.R.Firth (Linguist)
- Meaning of word in a text -> Context->set of words that appear around the word in a text
- Ex: Look at sentences using the word "Finance" to know it's meaning:

> Finance is the backbone of any thriving economy, orchestrating the flow of capital and investment.
>
> She excelled in her finance courses, mastering the art of managing budgets and analyzing financial statements.
>
> The company's finance department diligently monitored expenses and revenue to ensure fiscal stability.
>
> His passion for finance led him to pursue a career as an investment banker, navigating the complexities of the stock market with finesse.
>
> Understanding personal finance is crucial for individuals to achieve financial independence and security.

REPRESENTATION OF WORDS USING CONTEXT

WORD VECTORS

- Word Vector : Representation of word -> A dense vector is built such that word vectors of words appearing in it's context are similar to this word vector.
- 2 major types : Word2Vec and GLOVE
- Idea behind Word2Vec:
  - We possess an extensive text corpus, serving as a collection of words.
  - Each word within this text corpus needs to be denoted by a vector.
  - As we traverse through every position *t* in the text, we identify a central word *c* and its surrounding context words *o*.
  - By leveraging the similarity between the vectors representing *c* and *o*, we find the probability of *o* given *c* (or vice versa).
  - Continuously refining the word vectors of each word, aiming to maximize this probability.



WORD VECTORS

For each position $t = 1, \ldots, T$

Predict Content words given a centre word w_t.

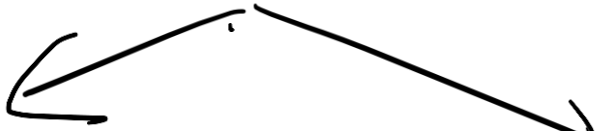
$$\text{likelihood} = L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \neq 0}} P(w_{t+j} | w_t; \theta)$$

$$\text{Loss} = J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log P(w_{t+j} | w_t; \theta)$$

WORD VECTORS

How to calculate $p(w_{t+j} | w_t; \theta)$?

Each word 'w' will have 2 vec per word



u_w (outside word)

v_w (centre word)

WORD VECTORS

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in Vocab} \exp(u_w^T v_c)}$$ } looks like Softmax func.

Softmax (x_i) gives prob. Score of x_i as p_i

$$p_i = \frac{\exp(x_i)}{\sum_{j=1}^{N} \exp(x_j)}$$

Soft $\Rightarrow$ gives little prob. to each x_i.

max $\Rightarrow$ Maximizes prob of more frequent events x_i.

WORD VECTORS

Parameter θ to be optimized:

$$\theta = \begin{bmatrix} v_{w_1} \\ v_{w_2} \\ \vdots \\ u_{w_1} \\ u_{w_2} \\ \vdots \end{bmatrix} \in \mathbb{R}^{2*d*V}$$

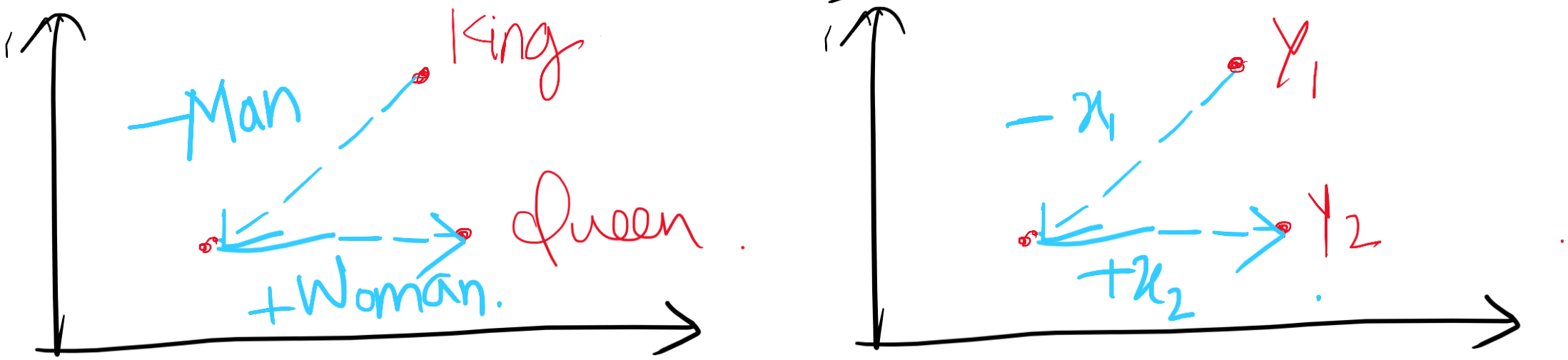
$d \rightarrow$ dim of each embedding

$V \rightarrow$ no. of words

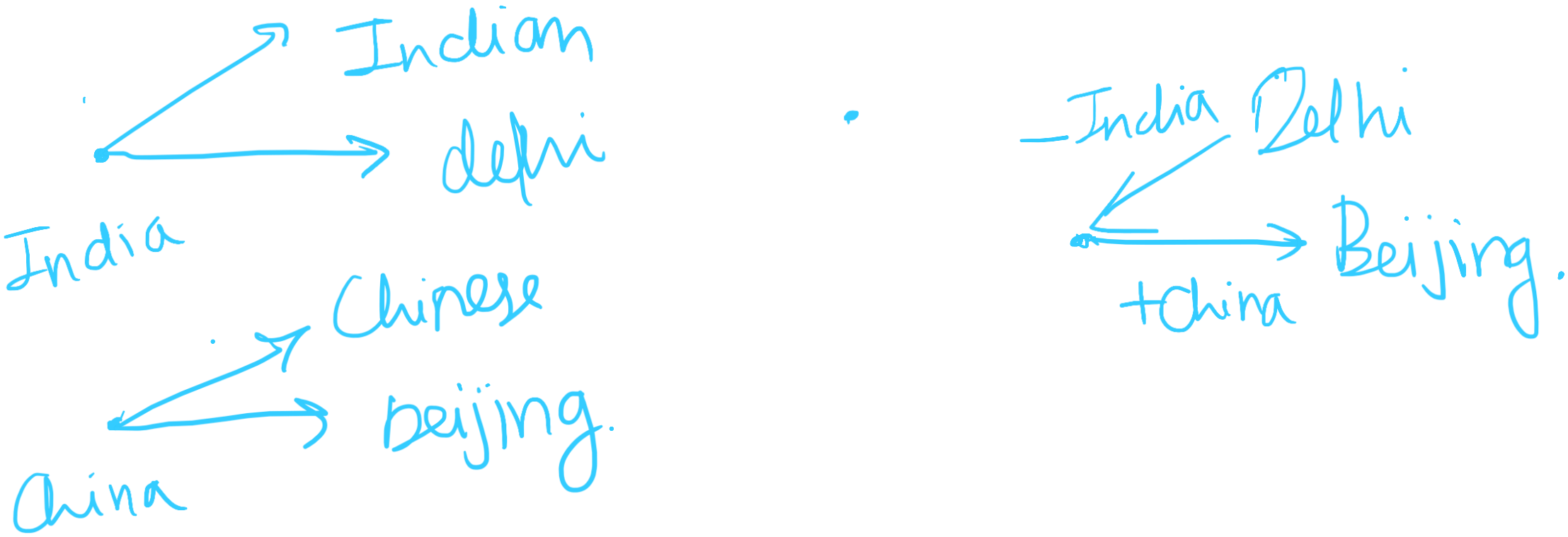
WORD VECTORS

Analogy → $x_1 : y_1 :: x_2 : y_2$

Given → x_1, x_2, y_1 ; Find → y_2 ?



WORD VECTORS



WORD VECTORS

WORD2VEC

- Why 2 word vecs? -> Easier optimization, since word was considered in 2 perspectives it was easier to optimize 2 vectors for both perspectives.
- Types:
  - SKIP GRAM (SG): Optimize p(outside|centre) irrespective of position of outside words
  - CONTINUOUS BAG OF WORDS (CBOW): Optimize p(centre|outside)

*We prefer SG over CBOW due to SG being less sensitive to overfit frequent words

- SG is optimized using **Negative Sampling**

Maximize prob $p(o|c)$ at each step 't' $\sim J_t(\theta)$

Overall objective fn to maximize $= \frac{1}{T}\sum_{t=1}^{T} J_t(\theta)$

$$J_t(\theta) = \log \sigma(u_o^T v_c) + \sum_{i=1}^{k} E_{j \sim P(\omega)} [\log \sigma(-u_j^T v_c)]$$

SKIP GRAM NEGATIVE SAMPLING

$$J_t(\theta) = \log \sigma(u_o^T v_c) + \sum_{k=1}^{K} [\log \sigma(-u_k^T v_c)]$$

Here $k = 1 \ldots K$ is drawn from a prob dist $P(\omega)$

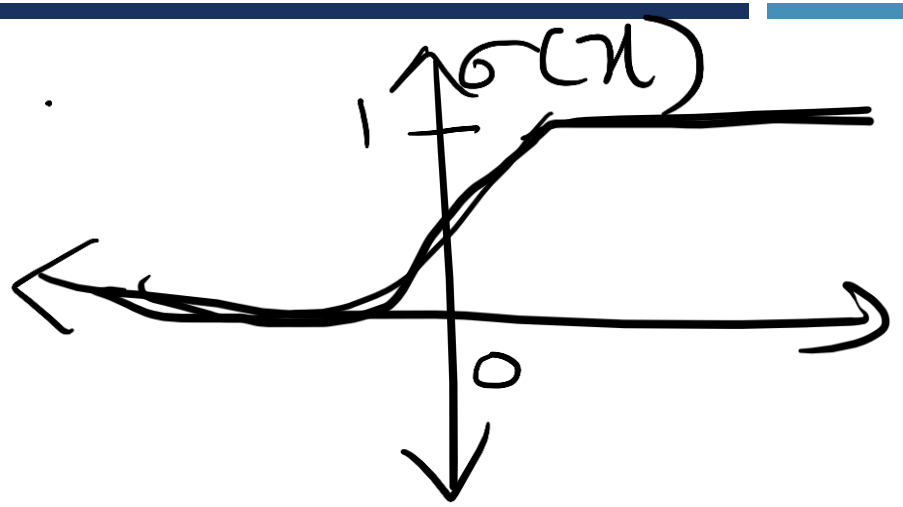
where $P(\omega) = \frac{U(\omega)^{3/4}}{Z}$

$U(\omega)$ → unigram dist of words in corpus

Raised to 3/4 helps sample less frequent words as well.

Z → Normalizing term

WORD VECTORS



$\log(u_o^T v_c)$

↓ Probability

$\sigma(\log u_o^T v_c)$

↓

↔ Prob of o & c occuring togethers.

maximize $u_o^T v_c$ → maximizing prob of o & c together

maximize $-u_j^T v_c$ → minimizing prob of c with other random words.

WORD VECTORS

QUESTIONS?

THANK YOU !